Tạp chí Tin học và Điều khiển học, T.30, S.1 (2014), 63–72

MÔ HÌNH NỀN TẢNG MÁY CHỦ CHIA SẺ VÀ BÀI TOÁN VECTOR PACKING TRONG CUNG CẤP TÀI NGUYÊN CHO DỊCH VỤ ẢO HÓA

PHẠM NGUYỄN MINH NHỰT[1], ĐOÀN VĂN BAN[2], LÊ VĂN SƠN[3]

[1]*Khoa Thương mại Điện tử, Trường Cao đẳng CNTT Hữu nghị Việt-Hàn*
[2]*Viện Công nghệ Thông tin, Viện Hàn lâm Khoa học và Công nghệ Việt Nam*
[3]*Khoa Tin học, Trường Đại học Sư phạm, Đại học Đà Nẵng*

Tóm tắt. Tối ưu hóa tài nguyên để cung cấp cho dịch vụ ảo hóa nhằm nâng cao hiệu suất dịch vụ IaaS (infrastructure as a service), đáp ứng yêu cầu khai thác tài nguyên hiệu quả trong Điện toán Đám mây là vấn đề đang được quan tâm hiện nay. Trong bài báo này, chúng tôi nghiên cứu bài toán cung cấp tài nguyên đa chiều từ nền tảng máy chủ chia sẻ cho dịch vụ ảo hóa, đưa ra công thức tính trên cơ sở bài toán quy hoạch tuyến tính nhằm tối thiểu hóa số máy chủ vật lý, độ phức tạp của bài toán và áp dụng các thuật toán chuẩn của bài toán vector packing để giải và đánh giá thông qua mô phỏng trên nhiều kịch bản thử nghiệm.

Từ khóa. Cung cấp tài nguyên, vector packing, điện toán đám mây, quy hoạch tuyến tính.

Abstract. In cloud computing, optimizing resource for virtual services to enhance IaaS service performance and meet the requirements of efficient resource exploitation is an attractive issue in recent times. In this paper, we study the problem of providing multi-dimensional resources based on shared hosting platforms for virtual services. We construct the problem as an optimization formulation that uses a linear programming to minimize the number of physical servers. The solution to this formulation is applying the standard algorithms of vector packing problem to solve and evaluate solutions via emulation-based program.

Key words. Resource allocation, vector packing, cloud computing, linear programming.

1. ĐẶT VẤN ĐỀ

Điện toán đám mây được xây dựng dựa trên thành tựu của nhiều lĩnh vực như kiến trúc hướng dịch vụ, tính toán lưới, ảo hóa,. . . trong đó, ảo hóa cho phép phân vùng tài nguyên của Y máy chủ vật lý ($Y \geq 1$) thành S máy ảo ($S \geq 1$) để thực thi các ứng dụng theo yêu cầu [1]. Hệ thống gồm nhiều máy chủ vật lý kết nối với nhau để chia sẻ các tài nguyên được gọi là nền tảng máy chủ chia sẻ [3]. Vấn đề cần quan tâm là tối thiểu hóa tài nguyên từ nền tảng đó để cung cấp cho dịch vụ ảo hóa, nhưng đảm bảo yêu cầu chất lượng dịch vụ QoS (Quality of Service).

Việc quản lý tài nguyên trong nền tảng máy chủ chia sẻ đã được các nhà chuyên môn quan tâm nghiên cứu, thể hiện trong các công trình [3, 4]. Tuy nhiên, các tác giả chưa nhấn mạnh đến việc phát triển các thuật toán cung cấp tài nguyên bền vững mà tập trung vào cách thức thực hiện của hệ thống. Chẳng hạn, Urgaonkar [3] đưa ra kỹ thuật thống kê việc sử dụng tài

nguyên và nhu cầu tài nguyên tối thiểu, Aron [4] trình bày công thức tính cho bài toán cung cấp tài nguyên có ràng buộc tối ưu nhưng giới hạn máy chủ là nguồn tài nguyên liền khối. Trong nội dung bài báo này, chúng tôi xem xét nhiều khía cạnh tài nguyên (tài nguyên vật lý) và sử dụng hàm mục tiêu tuyến tính được xác định dựa vào lý thuyết lập lịch công việc trong các tài liệu [6, 2, 10]. Bài toán cung cấp tài nguyên được xem xét trong cả hai trường hợp: tĩnh (static) và động (dynamic), nhưng sẽ tập trung giải quyết bài toán cho trường hợp tĩnh (nghĩa là nhu cầu tài nguyên không thay đổi).

Mặc khác, các vấn đề cung cấp tài nguyên có liên quan mật thiết đến bài toán vector packing và đã có nhiều công trình nghiên cứu về bài toán này, thể hiện trong các tài liệu [7, 8, 9, 11, 13, 14, 15]. Trong nội dung bài báo, sẽ dựa vào các thuật toán chuẩn của bài toán vector packing đã được công bố trong [8] để giải bài toán cung cấp tài nguyên cho các dịch vụ ảo hóa từ nền tảng máy chủ chia sẻ thông qua các kịch bản được xây dựng.

Những kết quả chính của bài báo có thể tóm tắt như sau:

1) Xây dựng mô hình cung cấp tài nguyên cho dịch vụ ảo hóa từ nền tảng máy chủ chia sẻ.

2) Phát biểu bài toán cung cấp tài nguyên từ nền tảng máy chủ chia sẻ dưới dạng bài toán quy hoạch tuyến tính và thiết lập độ phức tạp của bài toán.

3) Trên cơ sở bài toán vector packing, xây dựng thuật toán để giải quyết bài toán cung cấp tài nguyên nhằm tối thiểu số máy chủ vật lý.

Phần còn lại của bài báo được tổ chức như sau: Mục 2 dành để mô hình hóa bài toán cung cấp tài nguyên trong nền tảng máy chủ chia sẻ. Mục 3 giới thiệu sơ lược bài toán vector packing và các kết quả nghiên cứu về nó. Mục 4 trình bày giải pháp cho bài toán cung cấp tài nguyên của máy chủ chia sẻ và cuối cùng là phần kết luận cùng hướng phát triển.

2. CUNG CẤP TÀI NGUYÊN CHO DỊCH VỤ ẢO HÓA

Ảo hóa có thể được định nghĩa là sự trừu tượng các tài nguyên tính toán trong lưu trữ, thực hiện lệnh, tổ chức bộ nhớ, truyền thông trong mạng, . . . Hiện nay, công nghệ ảo hóa có thể kết hợp hoặc phân chia tài nguyên tính toán để biểu diễn trong một hoặc nhiều môi trường hoạt động. Kiến trúc phân tầng trong công nghệ ảo hóa được trình bày trong [1]. Trong đó, tầng ảo hóa đưa ra lịch trình, phân phối tài nguyên từ các máy chủ vật lý cho các máy ảo và làm cho mỗi máy ảo như nó hoàn toàn sở hữu tài nguyên này.

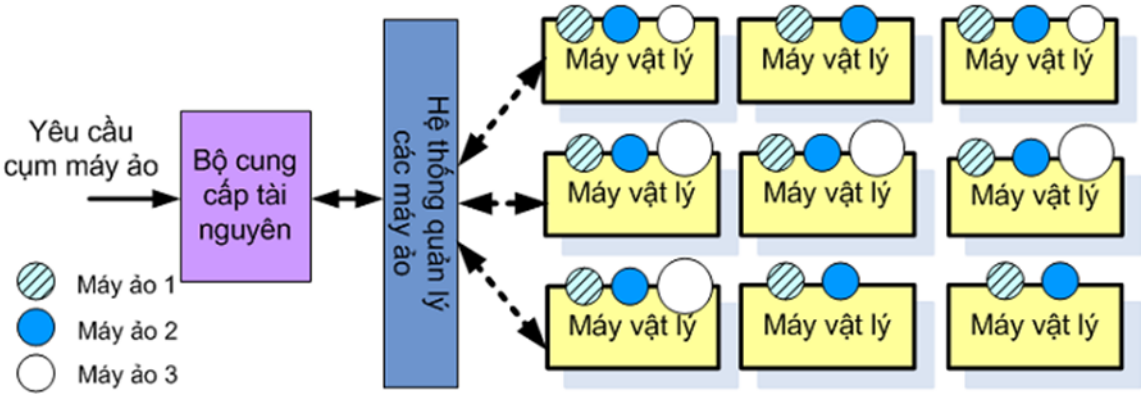
Trong mục này sẽ bày mô hình cung cấp tài nguyên dựa trên nền tảng máy chủ chia sẻ trong việc cung cấp tài nguyên (tài nguyên vật lý) cho dịch vụ ảo hóa; đưa ra một số khái niệm về tài nguyên, nhu cầu tài nguyên; công thức tính cho vấn đề cung cấp tài nguyên trên cơ sở bài toán quy hoạch tuyến tính với ràng buộc tối ưu và mục tiêu là tối thiểu hóa số máy chủ vật lý; thiết lập độ phức tạp của bài toán.

2.1. Mô hình cung cấp tài nguyên

Xét một nền tảng máy chủ chia sẻ đồng nhất gồm cụm các máy chủ vật lý có cấu hình giống nhau, được kết nối bằng thiết bị mạng tốc độ cao để chia sẻ tài nguyên nhằm cung cấp cho dịch vụ ảo hóa (xem Hình 1). Mỗi dịch vụ ảo hóa gồm có một hoặc nhiều máy ảo và hệ thống đảm bảo rằng các yêu cầu dịch vụ gửi đến máy chủ vật lý thích hợp.

Khi hệ thống nhận được yêu cầu cung cấp cụm máy (ảo) để thực thi các ứng dụng, hệ thống sẽ đáp ứng bằng cách thiết lập một hoặc nhiều máy ảo để thực thi yêu cầu đó. Các máy

ảo chạy trên máy chủ vật lý dưới sự quản lý của hypersivor [1] và tiêu thụ tài nguyên theo tỷ lệ khác nhau. Hệ thống quản lý các máy ảo có nhiệm vụ kiểm soát các hypersivor để xác định tỷ lệ tiêu thụ tài nguyên của các máy ảo. Cuối cùng, bộ cung cấp tài nguyên có nhiệm vụ ra quyết định từ chối hoặc đáp ứng các yêu cầu và phân chia tỷ lệ tài nguyên đến các máy ảo.



Hình 1. Hệ thống gồm một nền tảng máy chủ chia sẻ có 9 máy vật lý và cụm máy ảo có 3 loại máy ảo

Mục đích của nghiên cứu là xây dựng bài toán như một phần của bộ cung cấp tài nguyên và dựa trên các thuật toán chuẩn của bài toán vector packing được đưa ra trong tài liệu [8], đề xuất thuật toán để tìm số máy chủ vật lý tối thiểu dựa trên nhu cầu tài nguyên nhằm cung cấp cho dịch vụ ảo hóa.

2.2. Tài nguyên và nhu cầu tài nguyên

Để đáp ứng nhu cầu tài nguyên cho dịch vụ ảo hóa, mỗi máy chủ vật lý cung cấp một số loại tài nguyên, như: CPU, dung lượng RAM, băng thông I/O,... Trong thực tế, mỗi dịch vụ ảo hóa có hai loại nhu cầu tài nguyên: nhu cầu chặt và nhu cầu lỏng. Nhu cầu chặt biểu thị phần trăm cụ thể của tài nguyên yêu cầu, dịch vụ không hưởng lợi từ phần lớn hơn và không thể hoạt động với phần nhỏ hơn từ tài nguyên được cung cấp. Ví dụ, một dịch vụ có 2 nhu cầu chặt: nó yêu cầu 80% không gian RAM và 20% không gian đĩa của máy chủ. Nhu cầu lỏng biểu thị phần trăm tối đa của tài nguyên mà dịch vụ có thể sử dụng, dịch vụ không hưởng lợi từ phần lớn hơn nhưng có thể hoạt động với phần nhỏ hơn với chi phí giảm. Ví dụ, một dịch vụ có 2 nhu cầu lỏng: nó có thể sử dụng lên đến 40% băng thông I/O và lên đến 60% công suất CPU của máy chủ.

Tỷ số giữa phần trăm tài nguyên được cung cấp và phần trăm tài nguyên nhu cầu lỏng gọi là năng suất dịch vụ (NSDV). Ví dụ, khi dịch vụ có nhu cầu lỏng CPU là 60%, nhưng chỉ được cung cấp 30%, thì NSDV là $30/60 = 0.5$. Việc sử dụng tài nguyên đối với nhu cầu lỏng thường là quan hệ tuyến tính. Chẳng hạn, trong ví dụ trên, nếu dịch vụ được cung cấp 30% CPU (tức là chỉ một nửa so với nhu cầu) thì khả năng nó chỉ sử dụng 20% băng thông I/O (tức là chỉ một nửa so với nhu cầu). Điều này phù hợp với thực tế vì khi phần trăm công suất CPU cần cung cấp cho các ứng dụng giảm, dẫn đến tiêu hao tài nguyên khác cũng bị giảm (trong trường hợp này là băng thông I/O).

Như vậy, để đơn giản NSDV của tất cả nhu cầu lỏng có thể biểu diễn cùng một giá trị và giá trị của nó nằm trong khoảng 0 và 1. Trường hợp đặc biệt, nếu NSDV bằng 0 thì dịch vụ không được cung cấp tài nguyên (lỗi do thủ tục cung cấp tài nguyên), nếu NSDV bằng 1 thì

tài nguyên được cung cấp bằng với tài nguyên được yêu cầu. Tuy nhiên, cần xét đến trường hợp có ràng buộc NSDV theo quy định đáp ứng yêu cầu QoS, ràng buộc đó được biểu diễn bởi tích số giữa nhu cầu lỏng với yêu cầu QoS và được gọi là nhu cầu lỏng ràng buộc. Ta cũng giả định rằng nhu cầu chặt hoàn toàn độc lập nhu cầu lỏng, cung cấp tài nguyên cũng bị lỗi nếu nhu cầu chặt của dịch vụ không được đáp ứng.

2.3. Hàm mục tiêu và các ràng buộc

Giả định mỗi dịch vụ ảo hóa là một máy ảo riêng lẻ và có nhu cầu tài nguyên không đổi (trường hợp tĩnh). Ta xây dựng bài toán cung cấp tài nguyên (VSMSA) trên cơ sở bài toán quy hoạch tuyến tính với các biến hữu tỷ và nguyên. Xét S_i dịch vụ, với $i = 1, ..., n; S_i > 0$. Các cụm máy chủ có Y_j máy vật lý giống nhau, với $j = 1, ..., m; Y_j > 0$. Mỗi máy chủ cung cấp D_k loại tài nguyên, với $k = 1, ..., d$ và ma trận nhu cầu tài nguyên là

$$\textbf{Requirements Matrix} = \begin{pmatrix} r_{11} & r_{12} & ... & r_{1d} \\ r_{21} & r_{22} & ... & r_{2d} \\ ... & ... & ... & ... \\ r_{n1} & r_{n2} & ... & r_{nd} \end{pmatrix} \tag{2.1}$$

trong đó, phần tử r_{ik} biểu thị nhu cầu tài nguyên của dịch vụ thứ i với loại tài nguyên k và có giá trị giữa 0 và 1.

Ma trận cung cấp tài nguyên cho dịch vụ S_i từ Y_j máy chủ vật lý là

$$\textbf{Allocation Matrix} = \begin{pmatrix} x_{11} & x_{12} & ... & x_{1m} \\ x_{21} & x_{22} & ... & x_{2m} \\ ... & ... & ... & ... \\ x_{n1} & x_{n2} & ... & x_{nm} \end{pmatrix} \tag{2.2}$$

trong đó, phần tử x_{ij} là số nhị phân có giá trị 1 nếu dịch vụ i chạy trên máy chủ vật lý j và bằng 0 nếu ngược lại. Gọi α_{ik} là số nhị phân, bằng 1 nếu r_{ik} là một nhu cầu chặt, và bằng 0 nếu r_{ik} là một nhu cầu lỏng, β_{ij} biểu thị năng suất dịch vụ (NSDV) của dịch vụ S_i trên máy chủ Y_j, y_j số máy chủ vật lý để cung cấp tài nguyên cho dịch vụ i. Bài toán cung cấp tài nguyên được biểu diễn dưới dạng bài toán quy hoạch tuyến tính với các ràng buộc và hàm mục tiêu như sau

$$x_{ij} \in \{0, 1\}, \beta_{ij} \in Q, \ \forall i, j \tag{2.3}$$

$$\textstyle\sum_j x_{ij} = 1, \quad \forall i \tag{2.4}$$

$$y_j \geqslant x_{ij}, \quad \forall i, j \tag{2.5}$$

$$\textstyle\sum_i (\beta_{ij}(1 - \alpha_{ik}) + \alpha_{ik}) r_{ik} x_{ij} \leqslant 1, \quad \forall k, j \tag{2.6}$$

và hàm mục tiêu là $\min \sum_j y_j$. (2.7)

Ràng buộc (2.3) xác định miền của các biến. Ràng buộc (2.4) biểu thị trạng thái có một dịch vụ S_i chạy trên máy chủ Y_j. Ràng buộc (2.5) biểu thị trạng thái mà một máy chủ Y_j có được sử dụng hay không. Ràng buộc (2.6) biểu thị trạng thái mà tổng số phần trăm nhu cầu tài nguyên cho dịch vụ S_i luôn luôn nhỏ hơn hoặc bằng tổng số tài nguyên của máy chủ vật

lý Y_j, biểu thức trong phép lấy tổng cho thấy rằng: nếu nhu cầu tài nguyên r_{ik} là nhu cầu lỏng thì $\alpha_{ik} = 0$ và phần trăm tài nguyên D_k được sử dụng trên máy chủ Y_j là $\beta_{ij} \times r_{ik}$; nếu nhu cầu tài nguyên r_{ik} là nhu cầu chặt thì $\alpha_{ik} = 1$ và phần trăm tài nguyên D_k được sử dụng trên máy chủ Y_j là r_{ik}. Cuối cùng, ràng buộc (2.7) chính là hàm mục tiêu biểu thị số máy chủ vật lý để cung cấp tài nguyên cho dịch vụ ảo hóa, là tối thiểu hóa y_j.

2.4. Độ phức tạp

Để phân tích độ phức tạp của bài toán VSMSA, ta gọi bài toán quyết định của nó là VSMSA-Dec như sau: Có hay không việc gán S dịch vụ, mỗi dịch vụ có nhu cầu tài nguyên r_{ik} cho Y máy chủ ?

Định lý 1. *Bài toán quyết định VSMSA-Dec là NP-C.*

Chứng minh

(i) Dễ dàng nhận thấy bài toán VSMSA-Dec là NP. Bởi vì giải pháp nếu tồn tại có thể được kiểm tra trong thời gian đa thức.

(ii) Xét bài toán vector packing (VP) được trình bày trong [8, 9] là NP-C và được phát biểu như sau: Cho tập A gồm các phần tử là các vector d chiều được biểu diễn bởi bộ $d(a_1^i, a_2^i, ..., a_d^i)$ và tập B gồm các phần tử là các vector d chiều được biểu diễn bởi bộ $d(1, 1, ..., 1)$. Đặt các phần tử của tập A vào trong các phần tử của tập B sao cho $\sum_{i \in B} a_j^i \leqslant 1,\ \forall j = 1, ..., d$.

Bài toán VSMSA dẫn được từ bài toán VP như sau: Xét số máy chủ vật lý là B (tức là, $Y = B$), số dịch vụ là A (tức là, $S = A$), số loại tài nguyên là j (tức là $k = j$) và nhu cầu tài nguyên của dịch vụ i ứng với loại tài nguyên k là a_j^i (tức là $r_{ik} = a_j^i$). Với mỗi dịch vụ i, thiết lập $\alpha_{ik} = 0$ (tức là, chỉ xét nhu cầu lỏng, đối với nhu cầu cứng chứng minh tương tự). Rõ ràng, bài toán VP cung cấp một giải pháp cho bài toán quyết định. Ngược lại, một giải pháp cho bài toán quyết định của VSMSA sẽ cung cấp một giải pháp cho bài toán VP. Từ (i) và (ii) là điều chứng minh. ■

3. BÀI TOÁN VECTOR PACKING

Như đã trình bày trong mục 2.4, vector packing được xem như là bài toán lập lịch với tài nguyên hạn chế. Đây là bài toán NP-C, các phiên bản của thuật toán tham lam (First Fit, Best Fit, Next Fit) đã được Kou giới thiệu trong [8] và Maruyama giới thiệu trong [9], hành vi trong trường hợp xấu nhất, trung bình đã được nghiên cứu trong các tài liệu [7, 11]. Kết quả cho thấy rằng những thuật toán đó bảo đảm hiệu suất $d + \delta$ với d là chiều của vector, δ là một hằng số, $\delta \leq 1$. Hơn thế nữa, trong tài liệu [13], Fernandez de la Vega trình bày một thuật toán đảm bảo hiệu suất $d + \varepsilon$ với mọi $\varepsilon > 0$ bằng cách tái sử dụng thuật toán có hiệu suất đảm bảo $1 + \varepsilon$. Các đảm bảo này đã được cải thiện trong công trình [14], trong đó Chekuri đề xuất một thuật toán với độ phức tạp $O(\ln d)$ với hiệu suất đảm bảo $2 + \ln d$ cho d lớn. Gần đây, Bansal [15] đã đưa ra thuật toán với hiệu suất đảm bảo $1 + \ln d$.

Ý tưởng dựa trên các thuật toán giải bài toán vector packing để giải bài toán VSMSA, đó là đặt S_i *vector* D_k chiều vào Y_j (trong trường hợp này S_i là dịch vụ, Y_j số là máy chủ vật lý và D_k là loại tài nguyên). Trong phạm vi bài báo này, chỉ xem xét các thuật toán chuẩn: First Fit, Best Fit được đưa ra trong [8]. Trước khi áp dụng các thuật toán này để đặt các phần tử S_i vào Y_j, các phần tử S_i được sắp xếp theo thứ tự giảm dần theo 3 tiêu chuẩn, đó là:

- Từ điển (Lexicographical): Cho $k \geq 1,\ \theta^k = \{(S_1, S_2, ..., S_k),\ \forall i, 0 \leq S_i \leq 1\}$ và $a, b \in \theta^k.a \leq b$ khi và chỉ khi $a = b$ hoặc thành phần khác 0 đầu tiên của $b - a$ là số dương.

- Thành phần cực đại (Maximum Component): Cho $k \geq 1, \theta^k = \{(S_1, S_2, ..., S_k),\ \forall i, 0 \leq S_i \leq 1\}$ và $a, b \in \theta^k.a \leq b$ khi và chỉ khi thành phần cực đại trong b không nhỏ hơn thành phần cực đại trong a.

- Tổng cực đại (Maximum Sum): Cho $k \geq 1, \theta^k = \{(S_1, S_2, ..., S_k),\ \forall i, 0 \leq S_i \leq 1\}$ khi và chỉ khi tổng các thành phần của b không nhỏ hơn tổng thành phần của a.

4. GIẢI PHÁP CHO BÀI TOÁN VSMSA

4.1. Giải pháp đúng

Giải pháp đúng cho bài toán VSMSA thực hiện trong thời gian theo hàm số mũ. Ta sử dụng bộ giải công khai cho bài toán quy hoạch tuyến tính là lp-slover được đưa ra trong [12] để tính toán giải pháp đúng cho trường hợp bài toán VSMSA nhỏ (vài ba dịch vụ) trên máy tính đơn có bộ vi xử lý Intel Core Duo 1.86 GHz, RAM 2Gb trong thời gian dưới một giờ.

Trong trường hợp bài toán lớn (số dịch vụ lớn) giải pháp này không khả thi do độ phức tạp của bài toán. Do đó, nội dung sau đây sẽ sử dụng các thuật toán chuẩn của bài toán vector packing đã nêu ra trong Mục 3 để giải.

4.2. Giải pháp dựa trên các thuật toán chuẩn của bài toán vector packing

Dựa trên các thuật toán chuẩn của bài toán vector packing như đã trình bày trong Mục 3, có thể xây dựng thuật toán để giải bài toán VSMSA như sau:

Đầu vào: Tập các dịch vụ S_i (mỗi dịch vụ có các nhu cầu tài nguyên r_{ik} tương ứng với loại nhu cầu α_{ik}) và tập các máy vật lý Y_j với năng suất dịch vụ β_{ij}.

Đầu ra: Tập các máy vật lý Y_j tối thiểu được dùng (tương ứng $x_{ij} = 1$).

Các bước của thuật toán:

- Bước 1: Dựa trên nhu cầu tài nguyên r_{ik}, xây dựng vector S_i có D_k chiều với các phần tử là r_{ik}.

- Bước 2: Áp dụng các tiêu chuẩn sắp xếp, sắp xếp giảm dần các phần tử của vector S_i.

- Bước 3: Áp dụng thuật toán First Fit, Best Fit để đặt lần lượt các phần tử của vector S_i vào các máy chủ vật lý Y_j sao cho thỏa mãn điều kiện

$$\sum_i (\beta_{ij}(1 - \alpha_{ik}) + \alpha_{ik}) r_{ik} x_{ij} \leqslant 1, \quad \forall k, j.$$

- Bước 4: Nếu đáp ứng nhu cầu tài nguyên chưa hoàn tất, quay lại bước 3.

- Bước 5: Nếu đáp ứng nhu cầu tài nguyên đã hoàn tất thì đầu ra là tập các máy vật lý Y_j (đây chính là giá trị của hàm mục tiêu của bài toán VSMSA).

Như vậy, tổ hợp từ hai thuật toán tham lam First Fit, Best Fit cùng với 3 tiêu chí sắp xếp như Mục 3, ta có được 6 thuật toán, đó là: BestFitDesSum, BestFitDesMax, BestFitDesLex, FirstFitDesSum, FirstFitDesMax và FirstFitDesLex. Cố định D_k, các thuật toán này được thực hiện với độ phức tạp $O(D \log Y + DY)$.

Phương pháp thực nghiệm

Để đánh giá các thuật toán, ta tạo ra tập các mẫu thực nghiệm một cách ngẫu nhiên như sau: Xét S dịch vụ và Y máy vật lý với chiều tài nguyên D. Tương ứng với mỗi dịch vụ, số nhu cầu chặt là $D/2$ và số nhu cầu lỏng là $D/2$. Tất cả các nhu cầu tài nguyên được lấy mẫu từ một phân bố xác suất với trung bình μ và độ lệch chuẩn δ. Mỗi dịch vụ có ρ xác suất để có một yêu cầu QoS.

Ta giả định giá trị của các tham số như sau. Năng suất dịch vụ $\beta_{ij} = 0.5$, số dịch vụ $S = 32, 64, 128, 256, 512$ chiều tài nguyên $D = 6$ (trong đó, *Số nhu cầu chặt = Số nhu cầu lỏng = 3*), $\mu = 0.5,\ \delta = 0.25,\ 0.5,\ 1.0,\ \rho = 0.25,\ 0.5,\ 1.0$ và tất cả các yêu cầu QoS có giá trị là 0.5 (tức là một nữa nhu cầu lỏng của dịch vụ phải được đáp ứng). Thực nghiệm với các giá trị khác hoặc với các giá trị ngẫu nhiên cũng dẫn đến kết quả tương tự. Tương ứng với các giá trị đó sẽ có $1 \times 5 \times 1 \times 1 \times 3 \times 3 = 45$ kịch bản. Với mỗi kịch bản, có thể tạo ra 100 mẫu ngẫu nhiên, như vậy sẽ có 4500 mẫu dữ liệu đầu vào để đánh giá thuật toán. Các mẫu thực nghiệm được tạo ra lưu vào tập tin có cấu trúc như Hình 2.

<Số dịch vụ>
<Số nhu cầu lỏng>
<Số nhu cầu chặt>
<Giá trị QoS 1><Nhu cầu lỏng 1 của dịch vụ 1> <nhu cầu lỏng 2 của dịch vụ 1>…<nhu cầu chặt 1 của dịch vụ 1> <nhu cầu chặt 2 của dịch vụ 1> …
<Giá trị QoS 2><Nhu cầu lỏng 1 của dịch vụ 2> <nhu cầu lỏng 2 của dịch vụ 2>…<nhu cầu chặt 1 của dịch vụ 2> <nhu cầu chặt 2 của dịch vụ 2> …
…

Hình 2. Cấu trúc tập tin dữ liệu thực nghiệm

Với mỗi thuật toán, ta sử dụng hai thước đo, thước đo thứ nhất là số máy chủ vật lý tối thiểu tương ứng với 5 giá trị của số dịch vụ là $S = 32; 64; 128; 256; 512$. Thước đo thứ hai là thời gian thực hiện thuật toán tính bằng giây. Giá trị của hai thước đo được lấy trung bình từ 900 (tức là, $3 \times 3 \times 100$) mẫu thực nghiệm. Chương trình mô phỏng các thuật toán được thực hiện bằng ngôn ngữ C++ và thời gian thực hiện được đo trên máy tính đơn có bộ vi xử lý Intel Core Duo 1.86 GHz, RAM 2Gb.

Kết quả và nhận xét

Giá trị số máy chủ vật lý tối thiểu (giá trị hàm mục tiêu) tương ứng với số dịch vụ khi thực hiện các thuật toán được trình bày trong Bảng 1. Thời gian thực hiện các thuật toán tương ứng với số dịch vụ được trình bày trong Bảng 2.

Để trình bày trực quan về mối quan hệ giữa số dịch vụ với số máy chủ tối thiểu và thời gian thực hiện của mỗi thuật toán, ta biểu diễn các mối quan hệ thông qua đồ thị trong Hình 3 và Hình 4.

Căn cứ kết quả trên bảng số liệu và đồ thị cho thấy: thời gian thực hiện thuật toán và số máy chủ vật lý tối thiểu tỷ lệ thuận với số dịch vụ, thời gian thực hiện thuật toán tương

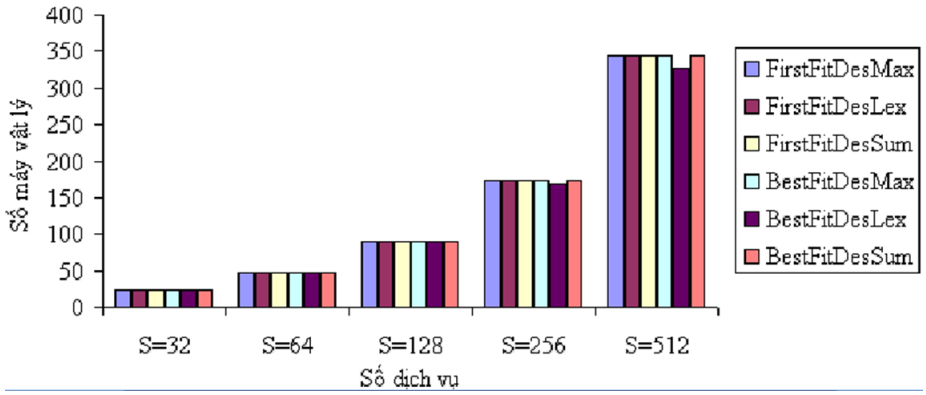
đối nhỏ và có thể áp dụng trong thực tế. Khi số dịch vụ nhỏ thì giá trị của hai thước đo này không khác biệt nhau nhiều giữa các thuật toán. Ngược lại, trong trường hợp số dịch vụ lớn ($S \geq 256$) thì có sự khác biệt rõ rệt, trong đó thuật toán BestFitDesLex cho số máy chủ vật lý tối thiểu thấp nhất còn thời gian thực hiện thuật toán FirstFitDesLex có giá trị ngắn nhất. Do đó, khi cung cấp tài nguyên cho dịch vụ ảo hóa (số dịch vụ lớn) mà quan tâm về thời gian thực hiện thì thuật toán FirstFitDesLex phù hợp, ngược lại khi quan tâm về số máy chủ vật lý tối thiểu thì thuật toán BestFitDesLex phù hợp.

Bảng 1. So sánh số máy vật lý tối thiểu khi thực hiện các thuật toán

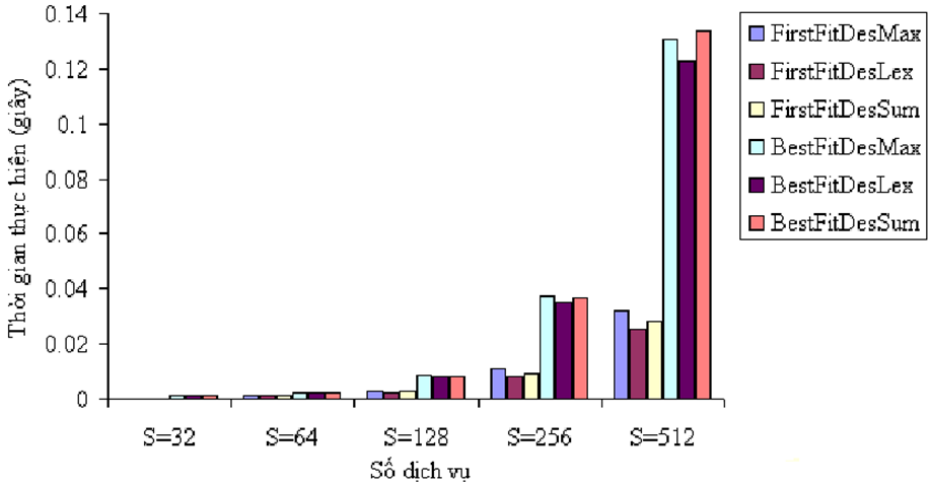
Tên thuật toán	Số dịch vụ				
	S = 32	S = 64	S = 128	S = 256	S = 512
FirstFitDesMax	24	47	90	174	344
FirstFitDesLex	24	47	90	174	344
FirstFitDesSum	24	47	90	174	344
BestFitDesMax	24	47	90	174	344
BestFitDesLex	24	47	89	170	327
BestFitDesSum	24	47	90	174	344

Bảng 2. So sánh thời gian thực hiện các thuật toán

Tên thuật toán	Số dịch vụ				
	S = 32	S = 64	S = 128	S = 256	S = 512
FirstFitDesMax	0.00009	0.00107	0.00303	0.01076	0.03193
FirstFitDesLex	0.00010	0.00114	0.00214	0.00827	0.02529
FirstFitDesSum	0.00016	0.00113	0.00268	0.00932	0.02791
BestFitDesMax	0.00121	0.00254	0.00833	0.03741	0.13107
BestFitDesLex	0.00123	0.00232	0.00783	0.03500	0.12253
BestFitDesSum	0.00128	0.00234	0.00800	0.03693	0.13380



Hình 3. Đồ thị biểu diễn mối quan hệ giữa số máy vật lý tối thiểu với số dịch vụ



Hình 4. Đồ thị biểu diễn mối quan hệ giữa thời gian thực hiện với số dịch vụ

5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Nội dung bài báo trình bày vấn đề cung cấp tài nguyên (tài nguyên vật lý) tĩnh, đa chiều dựa trên nền tảng máy chủ chia sẻ đồng nhất cho dịch vụ ảo hóa với ràng buộc tối ưu; mỗi dịch vụ là một máy ảo đơn lẻ; nhu cầu tài nguyên không đổi trong thời gian thực hiện và đảm bảo các yêu cầu QoS. Chúng tôi đã xây dựng công thức tính cho bài toán cung cấp tài nguyên VSMSA trên cơ sở bài toán quy hoạch tuyến tính với mục tiêu tối thiểu hóa số máy chủ vật lý; thiết lập độ phức tạp của bài toán; dựa trên ý tưởng từ các thuật toán chuẩn của bài toán vector packing trong tài liệu [8], đưa ra 6 thuật toán để cài đặt và đánh giá để giải bài toán VSMSA. Hướng nghiên cứu mở rộng và phát triển của đề tài là bài toán cung cấp tài nguyên động trong môi trường không đồng nhất cho dịch vụ ảo hóa, là hướng nghiên cứu cần thiết và cũng đang được các nhà chuyên môn quan tâm.

TÀI LIỆU THAM KHẢO

[1] Lê Văn Sơn, Phạm Nguyễn Minh Nhựt, Vấn đề cung cấp tài nguyên máy ảo trên cơ sở hạ tầng tính toán đám mây, *Tạp chí Khoa học và Công nghệ, Đại học Đà Nẵng* **5** (11) (2012) 63–71.

[2] A. Legrand, A. Su, and F. Vivien, Minimizing the stretch when scheduling flows of divisible requests, *Journal of Scheduling* **11** (5) (2008) 381–404.

[3] B. Urgaonkar, P. Shenoy, and T. Roscoe, Resource overbooking and application profiling in shared hosting platforms, *SIGOPS Operating Systems Review* **36** (SI) (2002) 239–254.

[4] M. Aron, P. Druschel, and W. Zwaenepoel, Cluster reserves: a mechanism for resource management in cluster-based network servers, *Proceedings of the 2000 ACM Sigmetrics International Conference on Measurement and Modeling of Computer Systems*, New York, USA, 2000 (90–101).

[5] W. Leinberger, G. Karypis, and V. Kumar, Multi-capacity bin packing algorithms with applications to job scheduling under multiple constraints, *Proceedings of the 1999 International Conference on Parallel Processing,* Aizu-Wakamatsu City, Japan, 1999 (404–412).

[6] M. A. Bender, S. Chakrabarti, and S. Muthukrishnan, Flow and stretch metrics for scheduling continuous job streams, *Proceeding SODA '98 Proceedings of the ninth Annual ACM-SIAM Symposium on Discrete Algorithms,* San Francisco, California, USA, 1998 (270–279).

[7] J. Csirik, J. B. G. Frenk, M. Labbe, and S. Zhang, On multidimensional vector bin packing, *Acta Cybernetica* **9** (4) (1990) 361–369.

[8] L. T. Kou, G. Markowsky, Multidimensional bin packing algorithms, *IBM Journal of Research and Development* **21** (5) (1977) 443–448.

[9] K. Maruyama, S. K. Chang, and D. T. Tang, A general packing algorithm for multidimensional resource requirements, *International Journal of Computer and Information Sciences* **6** (2) (1977) 131–149.

[10] Trịnh Thị Thúy Hằng, Lê Trọng Vĩnh, Hoàng Chí Thành, Nguyễn Thanh Thủy, Tối ưu đa mục tiêu trong việc lập lịch cho hệ thống tính toán lưới, *Tạp chí Tin học và Điều khiển học* **25** (1) (2009) 79–87.

[11] N. Roy, J. S. Kinnebrew, N. Shankaran, G. Biswas, and D. C. Schmidt, Toward effective multi-capacity resource allocation in distributed real-time and embedded systems, *Proceedings of the 11th Symposium on Object Oriented Real-Time and Distributed Computing*, Orlando, Florida, USA, 2008 (124–128).

[12] http://lpsolve.sourceforge.net/5.5/.

[13] W. Fernandez de la Vega and G. S. Lueker, Bin packing can be solved within $1 + \varepsilon$ in linear time, *Combinatorica* **1** (4) (1981) 349–355.

[14] C. Chekuri and S. Khanna, On multi-dimensional packing problems, *SIAM Journal on Computing* **33** (4) (2004) 837–851.

[15] N. Bansal, A. Caprara, and M. Sviridenko, Improved approximation algorithms for multidimensional bin packing problems, *FOCS '06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, Washington, DC, USA, 2006 (697–708).

Ngày nhận bài 29 - 10 - 2013
Nhận lại sau sửa ngày 20 - 01 - 2014